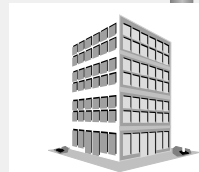


**Enterprise Architecture**

**Management Maelstrom**



*John A. Zachman  
Zachman International  
2222 Foothill Blvd. Suite 337  
La Canada, Ca. 91011  
1-818-244-3763*

© 2006 John A. Zachman, Zachman International

## **Introduction**

Enterprise Architecture presently appears to be a grossly misunderstood concept among management.

It is **NOT** an Information Technology issue.  
**It is an ENTERPRISE issue.**

It is likely perceived to be an Information Technology issue as opposed to a Management issue for two likely reasons:

- A. Awareness of it tends to surface in the Enterprise through the Information Systems community.
- B. Information Technology people seem to have the skills to do Enterprise Architecture if any Enterprise Architecture is being or is to be done.

© 2005 John A. Zachman, Zachman International

## Origins of Ent. Arch.

Frederick Taylor "Principles of Scientific Management" 1911

Walter A. Shewhart "The Economic Control of Quality of Manufactured Product" 1931

Peter Drucker "The Practice of Management" 1954

Jay Forrester "Industrial Dynamics" 1961

Peter Senge "The Fifth Discipline" 1990

Eric Helfert "Techniques of Financial Analysis" 1962

Robert Anthony "Planning and Control Systems: A Framework for Analysis" 1965

Sherman Blumenthal "Management Information Systems: A Framework for Planning and Development" 1969

Alvin Toffler "Future Shock" 1970

George Steiner "Comprehensive Managerial Planning" 1972  
Etc., etc., etc.

© 2006 John A. Zachman, Zachman International

## "Enterprise"

There are two implications to the word "Enterprise":

### I. Scope

The broadest possible boundary of the Enterprise.  
The Enterprise in its entirety.  
Enterprise-wide in scope.  
The whole thing.

### II. Content

ENTERPRISE Architecture is for ENTERPRISES.  
Enterprise Architecture has nothing to do with the Enterprise's systems or its information technology (except as they may constitute "Row 4" constraints).  
The end object is to engineer and manufacture the ENTERPRISE, NOT simply to build and run systems.

**"ENTERPRISE" ACTUALLY MEANS "ENTERPRISE"**

© 2005 John A. Zachman, Zachman International

# Caution

I hesitate to show you the next foil ...

for two reasons:

1. Some labels I put on some of the Cells may not suggest "primitive" models to some people.
2. I do not want to encourage anyone to change any of the words on the Framework graphic. I put some different labels on the Cells but I did not alter the classification schema nor did I modify the meta-models of any of the Cells.

The labels are meant to "elaborate" the idea of the Framework for a particular audience as a communications device. The labels are NOT the Framework. The Framework is actually the classification schema as expressed by the metamodel of the set of Framework Cells.

## The Framework for Enterprise Architecture - General Management Elaboration CLASSES OF DESCRIPTIVE REPRESENTATIONS

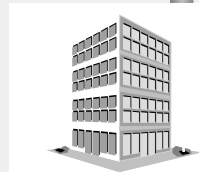
CLASSES OF ENTERPRISE PERSPECTIVES		WHAT	HOW	WHERE	WHO	WHEN	WHY	SCOPE
VISIONARIES	IMPORTANT RESOURCES LIST	IMPORTANT TRANSFORMATIONS LIST	IMPORTANT NETWORKS LIST	IMPORTANT ORGANIZATIONS LIST	IMPORTANT TIMINGS LIST	IMPORTANT MOTIVATIONS LIST		
MANAGERS	BUSINESS INVENTORY POLICY MODEL	BUSINESS PROCESS YIELD MODEL	BUSINESS LOGISTICS CAPACITY MODEL	BUSINESS WORK ALLOCATION MODEL	BUSINESS CYCLES TIMING MODEL	BUSINESS OBJECTIVES MODEL		BUSINESS
ANALYSTS	BUSINESS INVENTORY MANAGEMENT FILING SYSTEM	BUSINESS FUNCTIONALITY SCHEMATIC	BUSINESS LOGISTICS SYSTEM SCHEMATIC	BUSINESS ROLES AND RESPONSIBILITIES	BUSINESS SYSTEM PROCESSING CYCLES	BUSINESS RULE SYSTEM LOGIC		SYSTEM
TECHNOLOGISTS	FILING CONTAINER STORAGE MANAGEMENT	SYSTEMS DESIGN (MANUAL OR AUTOMATED)	LOCATION CAPACITY BLUEPRINT	WORK PRODUCT FORMAT DESIGN	BUSINESS SYSTEM TIMING DESIGN	BUSINESS RULE SYSTEM DESIGN		TECHNOLOGY
IDENTIFIERS	FILE IDENTIFICATION LISTINGS FOR PEOPLE OR MACHINES	SPECIFIC INSTRUCTIONS FOR MACHINES	LOCATION ADDRESSES FOR MACHINES	INDIVIDUAL WORK ASSIGNMENTS FOR PEOPLE OR MACHINES	TIMING SPECIFICATIONS FOR PEOPLE OR MACHINES	RULE SPECIFICATIONS FOR PEOPLE OR MACHINES		COMPONENT
RESOURCES	TRANSFORMATIONS	NETWORKS	ORGANIZATIONS	TIMINGS	MOTIVATIONS			OPERATIONS

Figure 1. The Framework for Enterprise Architecture (Business Vernacular Elaboration)

See CAUTION on previous foil.

**Enterprise Architecture**

## **The Framework and Management**



© 2005 John A. Zachman, Zachman International

## **The Framework and Management**

Management reasons for the **Columns**:

Column 1 has to do with inventory management.

Column 2 has to do with yield on transformations.

Column 3 has to do with capacity management.

Column 4 has to do with performance management.

Column 5 has to do with response times.

Column 6 has to do with plans and controls.

© 2005 John A. Zachman, Zachman International

## The Framework and Management

Management reasons for the Rows:

Row 1 has to do with setting Enterprise boundaries.

Row 2 has to do with defining Business Policies.

Row 3 has to do with institutionalizing the Business Policies (systematization).

Row 4 has to do with implementations (manual and/or automated).

Row 5 has to do with specific instructions (for people and/or machines).

Row 6 has to do with Enterprise operations.

## The Framework and Management

Oversimplifications and generalizations, however ...

Some reasons why the record keeping system does not:

- accurately reflect the actual inventories of resources or
- accurately reflect the financial characteristics of the Enterprise or
- provide consistent or accurate regulatory compliance (Sarbanes-Oxley type) information

are likely because:

1. The business policies that govern inventory management are not defined or not defined and managed consistently across the scope of the Enterprise. (Col.1, R 2)
2. The record-keeping system(s) (Col.1, Row 3) do not accurately and consistently reflect the business inventory policies. (Col. 1, Row 2)
3. The transaction data about resource acquisition and consumption is not accurately and "primitively" recorded. (Col. 2, Row 6; Col. 1, Row 6)

## The Framework and Management

Oversimplifications and generalizations, however ...

One reason why G & A and indirect expenses increase is likely because:

1. Compensation for inconsistencies in business policies regarding inventory management (Col. 1, Row 2) or in the filing system that records actual inventories. (Col. 1, Row 3.) (This is entropy ... compensation for disorder in the system.)

One reason why the yield on the business transformation of raw materials and energy to finished goods deteriorates over time is likely because:

1. The Business Processes (Col. 2, Row 2) and their supporting systems (Col. 2, Row 3) evolve (like the Winchester House evolved) ... they are not being engineered and have never been optimized.

© 2005 John A. Zachman, Zachman International

## The Framework and Management

Oversimplifications and generalizations, however ...

Some reasons why the distribution costs increase and network reliability decreases are likely because of:

1. Suboptimal positioning of storage and transmission capacities (Col. 3, Row 2)
2. Lack of standardization of locations and connections requiring "interfacing" or "translations." (Column 3, Rows 2 - 5)

Some reasons personnel costs increase are likely because:

1. Work product assignments are complex and overlapping and not clearly specified (Col. 4, Row 2)
2. Skills are not well-matched with the work assignments. (Col. 4, Row 2)

One reason why cycle times are excessive and difficult to predict is likely because:

1. Multiple cycles are interdependent and are interacting capriciously. (The law of unintended consequences.) (Col. 5, Rows 2 - 5)

© 2005 John A. Zachman, Zachman International

## The Framework and Management

Oversimplifications and generalizations, however ...

One reason why business objectives are difficult to realize is likely because:

1. Objectives are not defined such they change the state of some specific (single) thing that is within the Enterprise's control to change (Col. 6 and some Entity in Row 2). (Plans not attainable and/or not measurable.)

Some reasons why the Enterprise is not flexible are likely because:

1. No engineering has been done to separate the things that change independently from one another. (No Framework primitives.)
2. It's hard to figure out what to change ... or what changes will actually work. (No explicit Architecture.)

One reason why it is so difficult to take out costs is likely because:

1. There is no way to find recurring concepts that should or must be reused (i.e. no reusable primitive components.) (No Classification, i.e. no Framework.)

© 2005 John A. Zachman, Zachman International

## The Framework and Management

Oversimplifications and generalizations, however ...

The reason why there is:

- NO Integration - can't find recurring/reusable components
  - NO Interoperability - no standardization (of contents or containers)
  - NO Alignment - nothing explicit to which to align
  - NO Security - nothing to examine before implementation
  - NO Reduced time to market for systems implementations
    - nothing in inventory before you get the order
  - NO Flexibility - no separation of independent variables
  - NO Predictability - no knowledgebase
- Etc., etc., etc.

In short: There is NO ARCHITECTURE.

© 2005 John A. Zachman, Zachman International

## The Framework and Management

And ... life (business) is getting more complex ...

And ... the rate of change continues to increase ...

And ... once again, I submit ...

... Someday you are going to wish  
you had all these models  
(that is, Enterprise Architecture) ...

... and when that day comes,  
it's going to be too late to build them!

© 2005 John A. Zachman, Zachman International

## Lean and Mean

End Object: Minimum possible costs  
Minimum possible complexity

How do you do that?

Normalize EVERYTHING!

Remove ALL redundancy - NO recurring concepts

Redundancy:

1. Unnecessary costs of duplication - waste.
2. Compensatory costs of discontinuity - Entropy  
(Entropy = energy not available for productive work)
  - a. Internal costs - operating expenses
  - b. External costs - damage control, litigation

Second law of thermodynamics - the aging process.  
Over time, the energy required to support the system  
(entropy) increases. At the point in time the energy  
required to support the system exceeds the energy in the  
system, the system dies. How do you remove entropy?  
Re-engineer the system to remove disorder. Take out  
all discontinuous duplication. Engineer for simplicity.

© 2004 John A. Zachman, Zachman International

## Finding Redundancies

How do you discover recurring concepts?

How do you "normalize" anything? CLASSIFY.

But - the classification scheme has to be "clean." You can't have mixtures (apples and oranges) in any category because you won't be able to detect redundancies. The schema has to be "normalized" - one fact in one place.

And - the schema has to be comprehensive. You must have a category for every concept or you won't find the duplication of concepts that are not classified.

That is, the schema has to be comprised of single variable, "primitive" categories. No mixtures (composites.) The schema has to be complete, the total possible set of categories.

For example, the Periodic Table.

Anything less than the total set would either, by definition, be DE-normalized (contain composite categories) or could not accommodate the totality of the concepts.

© 2004 John A. Zachman, Zachman International

## The Framework

Primitive Models are architecture

Primitive models defined relative to the Enterprise are ENTERPRISE Architecture. Long term investments.

Composite Models are implementations

Composite models defined relative to one project are implementations. It is doubtful that you could define a composite, Enterprise-wide Model. It would be so complex, who could possibly understand it? Composite models are short term implementations.

**YOU DON'T HAVE TO NORMALIZE ALL 30  
PRIMITIVE MODELS TO REALIZE SHORT TERM  
OPTIMIZATION BENEFITS!**

(Note: discontinuity in some Columns may directly, negatively impact management's performance.)

**POINT NO. 2**

If you retain and maintain the primitive models, they are the baseline for managing change.

© 2004 John A. Zachman, Zachman International

# The Future

- A. Build Primitive Models
- B. Store Primitive Models
- C. Manage (Enforce) Primitive Models
- D. Change Primitive Models
- E. Assemble Composite Models  
from Primitive Models

It is not adequate merely to produce running code.  
(That was an Industrial Age idea.)

The long term Enterprise value  
lies in Enterprise "Engineering,"  
i.e. in the MODELS THEMSELVES!  
The "Knowledgebase" of the Enterprise  
(This is an Information Age idea!)

© 1990 John A. Zachman, Zachman International

## END STATE VISION

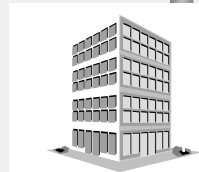
	What	How	Where	Who	When	Why	
Scope							Stonaries
Business							ecutive aders
System							chieets
Technology							ngineers
Components							Imple- menters
	Resources	Functions	Networks	Organizations	Timings	Motivations	

Some day  
You are going to wish you had  
all these models made explicit,  
Enterprise-wide,  
horizontally and vertically integrated,  
at excruciating level of detail !!!

© 1990 John A. Zachman, Zachman International

**Enterprise Architecture**

## **Enterprise Engineering Design Objectives**



© 2000 John A. Zachman, Zachman International

## **Engineering Design Objectives**

Alignment  
Integration  
Flexibility  
Interoperability  
Reduced Time-to-Market  
Quality  
Seamlessness  
Adaptability  
User-Friendliness  
Usability  
Reusability  
Security  
etc., etc.

Note: All of these desirable characteristics of systems (i.e. of the Enterprise) PRESUME the existence of Architecture (i.e. "primitive" models) and further, that the models were designed with those characteristics specifically in mind.

If no Architecture ("primitive" models) exist, these characteristics are simply "platitudes."

© 2000 John A. Zachman, Zachman International

## Alignment

"Alignment" means ...

... you want the implemented systems (Row 6)  
to align with the Enterprise intent (Row 1/2 Models).

Similar Manufacturing concept: "Quality"

"Alignment" would be the Enterprise equivalent of

"Total Quality Management."

© 2000 John A. Zachman, Zachman International

## Alignment

If you REALLY want the implemented systems (Row 6)  
(Enterprise) to be aligned with management's intent  
(Row 1/2), here is how you do it:

First, build Row 1 models.

Next, build Row 2 models.

Next, build Row 3 models.

Next, build Row 4 models.

Next, build Row 5 models,

ensuring the intent of each Row is successfully  
represented in the succeeding Row.

Next, compile and run.

Then, the implemented Enterprise will align with  
the Business Purpose.

It looks to me like any other alternative would require  
either a suspension of the laws of nature  
or, an inordinate amount of pure luck.

© 2000 John A. Zachman, Zachman International

## Alignment

Next question:

How do you intend to **KEEP** the implementation (Row 6) "aligned" with Management's intent (Rows 1/2) when everything is changing like gangbusters????:

Management's Intent (Rows 1/2)

The Design state of the art (Row 3)

The Technology constraints (Row 4) and

The Technology products (Row 5)  
and

The Enterprise Data (Column 1)

The Processes (Column 2)

The Distribution Network (Column 3)

The Organization (Column 4)

The Dynamics (Column 5) and

The Business Strategy (Column 6)

**Configuration Management**

**(You have to retain and maintain the models.)**

**"Someday, you are going to wish ..."**

© 2000 John A. Zachman, Zachman International

## Integration

"Integration" means ...

... you want no discontinuity between the various related concepts within the Enterprise. For example:

- **Scope Integration** - no discontinuity within any one kind of model across the scope of the Enterprise. (Internal sharing, the antithesis of "stovepipes"... efficiency.)
- **Horizontal Integration** - no discontinuity across the different kinds of related models from Column to Column. (Horizontal sharing - coordinated operations and change ... effectiveness.)
- **Vertical Integration** - no discontinuity between the various Rows, the Owner's, Designer's and Builder's constraints. (The end result is consistent with the requirements ... quality. This is "alignment.")

© 2000 John A. Zachman, Zachman International

# Integration

**Integrate:** To make into a whole by bringing all parts together; unify. (American Heritage Dictionary)

My paraphrase is "get rid of all the redundancy, discontinuity, inconsistency, disorder ... entropy ... make it all one coherent, optimized thing."

"Integration" implies sharing, NORMALIZATION. You have any one concept defined only one time. Any time you need some one concept, you reuse what has already been defined. You DON'T re-create that same concept creating redundancies, potential inconsistencies, discontinuities, etc. See "Lean and Mean" foils.

Similar Concepts:

- Seamlessness (Col. 2),
- Reusability (implied).

Reusability is the Enterprise equivalent of  
Standard Interchangeable Parts.

© 2000 John A. Zachman, Zachman International

ENTERPRISE ARCHITECTURE - A FRAMEWORK™

	What	How	Where	Who	When	Why	
Scope							Visionaries
Business		<b>Horizontal Integration (Any Row)</b>					Executive Leaders
System							Architects
Technology		<b>Scope Integration (Any Cell)</b>					Engineers
Components							Implementers
	Resources	Functions	Networks	Organizations	Timings	Motivations	

© John A. Zachman, Zachman International

© 2000 John A. Zachman, Zachman International

## Reusability

Reusability is related to Integration ...  
(Standard Interchangeable Parts)

Reusability is so easy to understand in physical objects.  
e.g. If you want to screw a carburetor on more than one  
(different kind of) engine, you have to engineer it to fit  
on whatever kinds of engines you want to screw it on  
BEFORE you get it (them) manufactured.

That is, if you want to engineer something to be reused,  
you have to know the total set of possibilities in order to  
engineer characteristics that will accommodate multiple  
uses.

Manufacturing Quality Assurance would test any one  
variable against an independent variable (for example,  
test the Enterprise Semantic Structure against the  
Enterprise Business Process Model. If the Semantic  
Structure supports the total set of Business Processes at  
any given point in time, it will likely support any Business  
Process for the foreseeable future as long as you don't  
materially change the nature of the Business.)

© 2000 John A. Zachman, Zachman International

## Reusability (cont)

The key to Reuse (and Integration) is Enterprise-wide,  
normalized, horizontally-integrated, PRIMITIVE models.

If you don't know the total set ...

... that is, if you don't have an Enterprise-wide, primitive  
model, and if you haven't tested it against at least one  
other horizontally-related, Enterprise-wide primitive  
model in the same Row ...

and it turns out that whatever you manufactured can be  
reused, it is only by accident, a miracle ... or else whoever  
wanted to use whatever it is, changed their use to fit  
whatever you manufactured.

Note: I think the probability of engineering reuse or  
integration of composite models is limited because a  
composite model can be reused only when the composite  
variables occur in exactly the same structure ... and the  
more variables there are in the composite, the less the  
probability they will occur in exactly the same structure.

© 2000 John A. Zachman, Zachman International

## Post - Integration

**Interfacing** (Column 1 word.)

You can do some after-the-fact "transformations" as long as they are cosmetic, that is, relate to format, or name, or instances only. You can't add content (create something out of nothing) or change structure.

("Middleware": Column 3 word for the same idea.)

**Interfacing** (generic word)

Hard-binding two independent variables -  
loss of flexibility.

Maintenance problem -  
number of connections between "n" points is

$$\frac{n^2 - n}{2}$$

It is a good "point-in-time" (short term) solution -  
good as long as there are few points and nothing changes.

© 2000 John A. Zachman, Zachman International

## Interoperability

I'm not too sure I know what "Interoperability" means:

Maybe it means want to run the same applications on different hardware and systems software ("portability").

Maybe it means you want the data to mean the same thing in two different implementations ("reusability").

Maybe it means that you want to share applications components between two different Enterprises ("integration").

Maybe it means all of the above ...

Interoperability = Portability + Reusability + Integration

I have already addressed Reusability and Integration so all that is left is Portability.

Maybe it means "Federated Architecture" - See Complex Enterprises.

© 2000 John A. Zachman, Zachman International

## Portability

**Portability:** If you REALLY want to run the same thing on different systems ...

- If the "systems" are all manufactured by the same manufacturer and are running the same operating system, portability is absolute.
- As soon as you introduce a different manufacturer and/or operating system, portability degrades.
- "Heterogeneous interoperability" (Portability + Reusability + Integration) is an oxymoron.
- Heterogeneous means you optimize the parts at the expense of the whole.
- Interoperability means you optimize the whole at the expense of the parts.
- Interfacing (gateways, middleware, etc.) mapping (many to many) independent variables increases maintenance liability to "m times n" and inhibits change.
- "Open Systems" is a great idea ...  
but it is not magic!

(See "Open Systems.")

© 2000 John A. Zachman, Zachman International

## Open Systems??

"Open Systems" says:

Somebody (vendor or user) is going to write code multiple times such that a given design (set of row 5 models) will run:  
under more than one operating system,  
on more than one computer,  
connected to more than one kind of line,  
under more than one network operating system,  
and using more than one data base management system.

Or else ... the user will have to build the logical models (row 3 models) and somebody (user or vendor) will write the code to dynamically (at run time) transform the row 3 models to row 4/5.

Or else ... everybody has to use a standard computer, operating system, line protocol, network operating system and data base management system.

Or else, the vendors will have to build their hardware, systems software, networks and data bases to standard specifications.

Or else ... somebody (vendor or user) will have to build and *maintain* a bunch of custom, gateway/conversion interfaces.

(In this last case, the question is going to become, "how many computers, operating systems, line protocols, network operating systems and database management systems can one organization (or vendor) afford to support in a changing environment?")

If you can't explain to me exactly how you intend to do this, I know I am going to have to depend on a miracle to happen.

© 2000 John A. Zachman, Zachman International

## Integration In Summary

**Integration:** If you REALLY want "Integration", that is if you do not want any discontinuity (Entropy) in the Enterprise ... if you want reusability, interoperability, seamlessness, standard interchangeable parts ...

... it has to be engineered into the product at whatever the desired scope of integration is at the outset, that is,  
**BEFORE**  
anything is implemented.

How do you achieve integration?

... not by accident,  
... not with hardware or software,  
... not by wishful thinking, or by announcement,  
... not after-the-fact,  
**ONLY by ENGINEERING!!**

**"Someday, you are going to wish ..."**

© 2000 John A. Zachman, Zachman International

## Flexibility

Flexibility means ...

... you want things implemented such that they can be changed in minimum time with minimum disruption, at minimum cost.

Note: NOT ZERO time, ZERO disruption, ZERO cost!  
i.e. not "Free, Perfect and Now"  
(book by Robert Rodin, CEO, Marshall Industries)

(Similar Concept: Adaptability)

("Flexibility" is closely related  
to "Change Management"  
and somewhat related to "Integration.")

© 2000 John A. Zachman, Zachman International

## Flexibility No. 1

If you REALLY want flexibility, for starters it would be helpful to be working with "standard, interchangeable parts," that is, "integrated" architecture.

That is, it would be helpful to have "normalized" models ... any one concept exists only one time ... so if you need to change it ... you change it once and it is changed for every deployment. You don't re-create it and create redundancies with the inherent inconsistencies and discontinuities and insurmountable maintenance problems.

Next, it would be helpful to have an inventory of all the reusable concepts so you knew where they were and what they were and could find them when you wanted to re-use them. (That is, in a "database.")

I already talked about all of this under the heading:  
"Integration."

© 2000 John A. Zachman, Zachman International

## Flexibility No. 2

**Flexibility:** If you REALLY want to change things in minimum time, with minimum disruption and cost ...

- Decouple (separate) the independent variables. (e.g. "Data Division" and "Procedure Division") You potentially can change one thing without having to change everything.
- Every ("*primitive*") cell of the Framework is an independent variable.
- "3 - Schema" as a concept is the mapping of two independent variables to a common, "conceptual schema" reducing the change liability to "m plus n."
- If you want to change the Enterprise with minimum time, disruption and cost, you have to retain and maintain the architectural *primitives*. (e.g. If you want to dynamically change the Row 4 technology constraints you have to manage the architecture at the Row 3 logical level.)

© 2000 John A. Zachman, Zachman International

## Managing Change

How do you change anything?

7,000 years of experience with older disciplines suggest that to change anything, you start with the drawings, the functional specs, the bills-of-material ...

How do you change buildings?

You start with the drawings, the functional ...

How do you change airplanes?

You start with the drawings, the functional ...

How do you change that PC on your desk?

You start with the drawings, the functional ...

If you DON'T HAVE the drawings, functional specs ...

You have 3 options:

1. Change by trial and error ... **HIGH RISK**
2. Re-create the drawings, functional specs ...  
**TAKES TIME AND COSTS MONEY**
3. **DON'T CHANGE IT!** (Scrap the whole thing and start over again.)

© 2000 John A. Zachman, Zachman International

## Flexibility - Summary

In summary, if you REALLY want flexibility:

1. Normalized concepts - any one concept only exists one time in the Enterprise. Anytime it is changed, it is changed once.
2. "Primitive" models are **KEY** - separate out the independent variables. You can change one concept without having to scrap and rework the entire implementation. (Each Cell is an independent variable.) Also, you can create an "infinite" number of composites from a finite set of primitives.
3. Repository - retain and maintain the "primitive" models to serve as a baseline for managing change.

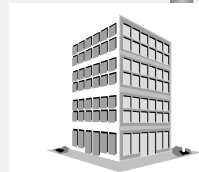
Note: If you have no architectural representations, you have no separation of independent variables, no baseline for managing change and therefore, *NO FLEXIBILITY* (all independent variables are implicit and imbedded in a unitary implementation ... the running system.)

**"Someday, you are going to wish ..."**

© 2000 John A. Zachman, Zachman International

**Enterprise Architecture**

## **Reducing Time-To-Market**



© 2000 John A. Zachman, Zachman International

## **Reducing Time-to-Market**

How do you reduce the time required to get to implementation (Row 6)?

1. Don't build any models.  
"You start writing the code ... "
2. Reduce scope - Build : "slivers"  
(Raises the "integration" question.)
3. Reduce level of detail - Make assumptions  
(Raises the "alignment" issue.)
4. Automate the process - "CASE" Tools.  
(Better, Faster, Cheaper than people.)
5. Concurrent Engineering - "JAD"  
(Parallel Process vs Sequential Process)
6. Buy packages - Prefabricated solutions.  
(Presumes architectural "fit.")
7. Assemble-to-order - The "asset paradigm."  
(Presumes a managed inventory of models.)
8. Pray - If I were you, I would pray too.  
(We need all the help we can get!)

Can you think of anything else???

© 2000 John A. Zachman, Zachman International

## Reducing Time-To-Market

If you REALLY want to reduce the time it takes from when you discover you need a new system until it's operational:

1. If you wait until you get the order before you start to engineer and manufacture, you can only reduce time-to-market by reducing size/complexity of the product. Reduce scope - simplify - proliferate legacy problems.
2. If you have finished goods in inventory before you get the order, you can reduce time-to-market to virtually zero providing you change the use of the product to fit the product. Packages - implement AS IS. Change the Enterprise to fit the package.
3. If you have prefabricated parts in inventory that are designed to be assembled into more than one product, you can reduce time-to-market to the time it takes to pick the parts and assemble them to order. Virtually zero. What do you think has to be in inventory before you get the order? "Someday, you are going to wish .... "

© 2000 John A. Zachman, Zachman International

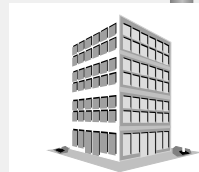
## Ent. Eng. Design Obj. - Summary

I simply do not believe it is possible  
to achieve any of these  
Enterprise Engineering Design Objectives  
without building primitive models,  
i.e. single abstraction by single perspective (single Cell)  
"PRIMITIVE" models.

© 2000 John A. Zachman, Zachman International

## Frequently Asked Questions

# Value Proposition



© 1990 John A. Zachman, Zachman International

## Value Proposition for Architecture

### A. Alignment

The implemented enterprise reflects the intent of "the Executive Leaders."

(In manufacturing - this equates to "Quality" - "TQM")

### B. Integration

The data means the same thing to everyone. Messages are successfully (and consistently) transmitted from node to node.

Everyone understands the objectives/strategy. (The enablers of "empowerment.")

(This is standard, interchangeable parts.)

### C. Change Management (Flexibility)

Independent variables - baseline for managing change. Retained, accumulated, Enterprise knowledge .

(Change with minimum time, disruption and cost.)

(This is "effectivity," change management.)

### D. I/S Responsiveness

Architecture plus "assemble-to-order" processes.

(This is reduced "time to market" - "Mass Customization.")

© 1990 John A. Zachman, Zachman International

## Value Proposition for Architecture

### E. Security

Until now, "amateurs" ... (2004 & beyond) professional types of attackers, targeting crucial online systems.

Robert Clyde CTO, Symantec, 2003

... today's threat ... well-organized criminal syndicates employing sophisticated and structured attack techniques.

World Bank 2003

**Roger Schell, AESec Corp. roger.schell@aesec.com**

Dependent on computer tech. to prevent grave damage

Software of uncertain pedigree

Commercial software & open source easily subverted

Much overseas - software supplied by our enemies

Sometimes called the MLS (MultiLevel Security) problem

Prevents interconnection of networks

Impedes real-time sharing of information

**Subversion** demonstrated as the attacker's tool of choice

### Architecture for High Assurance MLS -

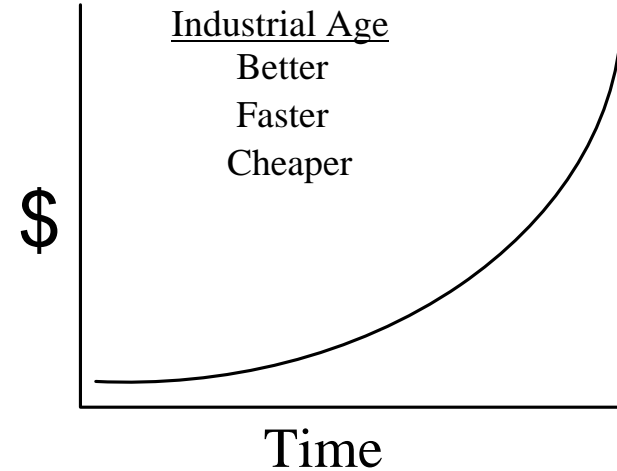
### Major Issues for Verifiable Protection

Presentation at StratCom 4-6-2004

*I submit - there is NEVER going to be verifiable protection for high assurance without EA! Some day ... !!!*

© 2000 John A. Zachman, Zachman International

## Short Term Investment



Expense-based, short term oriented, implementation-dominant, "cost-justified," "you start writing the code ... I'll go find out what the users have in mind."

Start manufacturing before you do any engineering.

**NO ARCHITECTURE**

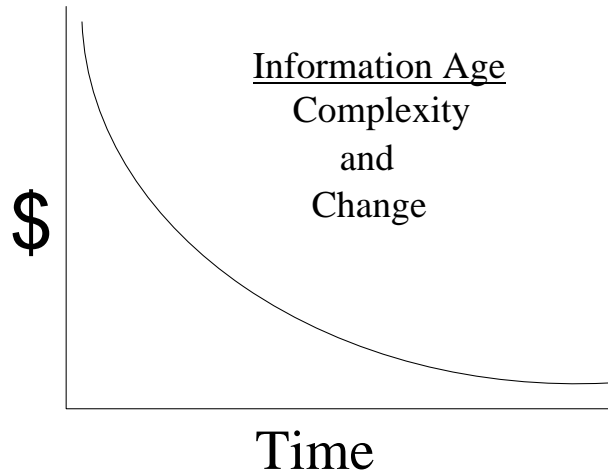
Quick implementations. Consumables.

Point-in-time solutions. One time use.

"Pay me now or pay me later" - Scrap and rework.

© 2000 John A. Zachman, Zachman International

# Long Term Investment



Long term, asset-based, integration and normalization dominant, investment-oriented, engineered for reusability.

Do the Engineering BEFORE you start manufacturing.

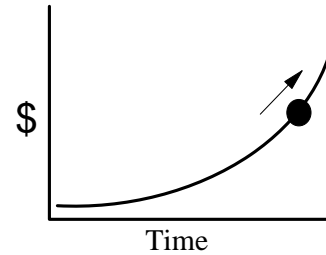
## ARCHITECTURE

Create re-usable assets. Minimize scrap and rework.

Takes up-front time and money for initial investment.

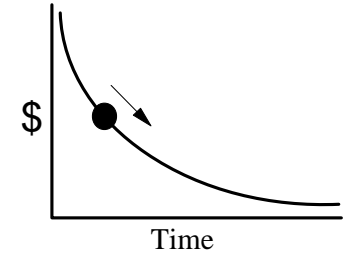
© 2000 John A. Zachman, Zachman International

# Engineering Design Objectives



This is the short term value:

Implementation



These are long term values:

- Alignment
- Integration
- Flexibility
- Interoperability
- Reduced Time-to-Market
- Quality
- Seamlessness
- Adaptability
- User-Friendliness
- Usability
- Reusability
- etc., etc.

© 2000 John A. Zachman, Zachman International

## Cheaper and Faster

Using a top-down, Enterprise Architecture approach, a rigorous, enhanced Information Engineering Method, Three-Schema Data Architecture and CASE technology:

Cost per new entity type/RDBMS table  
reduced from >\$150,000 to <\$10,000.

Enterprise data handling labor reduced 50%.

Reduced development time of 25% through  
improved communication and conflict resolution.

Development time and cost reductions for every  
succeeding implementation of >50%,  
compounded, through reuse of database and  
application components with no modifications.

Reduced disk space for data (including history)  
of 20% - 80% through elimination of redundancy.

*Reference: Doug Erickson 614-751-5078*

© 2004 John A. Zachman, Zachman International

## Cheaper and Faster

State of Ohio: Workers Compensation Board

Rates System 1,030 entity types  
(operational - 2 1/2 years elapsed time)

Benefits Payments 720 e/t's (Reused 470)  
(operational)

Retro Rated Billing 230 e/t's (Reused 220)  
(operational)

4 years total elapsed time, no database failures,  
< 40 hours required maintenance

Health Provider Mgmt 415 e/t's (Reused 255)  
(under development)

Total Cost per entity type \$25,000 (conservative)  
includes legacy data cleansing  
all data conversion costs  
all interfaces with remaining legacy  
no redundancy - reduced entropy  
complete Enterprise alignment and integration

Total savings: 945 (reused) x \$25,000 = \$23,625,000

© 2004 John A. Zachman, Zachman International

## Cheaper and Faster

Recent Package implementation: \$50,000/ET (conserv.)  
 (no data cleansing, no data conversions, no legacy  
 interfaces, added redundancy and 60% functionality)

Recent Custom Apps: \$100,000- \$150,000/Ent. Type  
 typical legacy, redundant environment (re: entropy)

### Comparative Costs

Trad. Appln Devel 2395 e/t's x \$140,000 = \$335,300,000  
 Package 2395 e/t's x \$50,000 = \$119,750,000  
 Ent. Arch. 2395 - 945 e/t's x \$25,000 = \$36,000,000  
 (and Enterprise Architecture approach  
 "aligned," low maintenance, no entropy)

Re: "Reusable code"

In three operational Systems:

6,128 action blocks Avg. Reuse factor = 7.0  
 (Trad. A/D: Code, test, maintain 42,896 subroutines)  
 (attributable to granularity and precision of the data  
 model, i.e. many processes use the same data.)

Reference: Doug Erickson 1-614-751-5078

© 2004 John A. Zachman, Zachman International

## Cheaper and Faster

### State of Ohio

### Different State

Workers Comp.	<i>Application</i>	Child Welfare
IEF/CoolGen	<i>Same CASE Tool</i>	IEF/CoolGen
Architected	<i>Different Methodology</i>	Classic
1030	<i>Entity Types</i>	300
2.5 Years	<i>Elapsed Time</i>	12 Years
-	<i>Development Costs</i>	\$42 Mil.
\$25,000	<i>Cost per Entity Type</i>	\$140,000
		(2 prime contractors and one local cntrtr. Estimating 3 more years to enhance/fix)
		Reference: Doug Erickson 614-751-5078

© 2004 John A. Zachman, Zachman International

## The REAL Benefit

From: Jim Tompkins (**Large Customer**)

To: Lauree Raica (**Chief Risk Officer**)

THANK YOU! THANK YOU!! THANK YOU!!!

This is great news for Frank Gates and The Service Assn. of Ohio. I appreciate so much your continuing efforts to help facilitate the improved turnaround time on these quarterly claim cost updates. I also want to express my appreciation to the "systems staff" at BWC in moving us forward with receiving the claims status and effective date information in the updated PIRS system. This will be a significant benefit.

From: Jim Romig (**Chief, Employer Relations**)

To: Admiral Jim Conrad (**CEO**)

Adm. Conrad, the IT Department did a great job in speeding up the turnaround time for quarter ending reserves being posted on Dolphin. What used to take 24 days now takes less than 10 . We have received several thank you's from TPAs since they are now able to get Sept. 30th data by Oct. 10th instead of Oct. 24th.

© 2004 John A. Zachman, Zachman International

## The REAL Benefit

From: Leo Genders (**Dpty. Mgr. Appln. Dvlpmnt.**)

To: Rates & Payments Project team

This is excellent news and worthy of high praise. Great job to all involved. It is not often that an outside customer praises the internal IT Team!

From: Kevin Ribble (**Appln. Dvlpmt. Mgr.**)

To: Nary Loganathan, Russel Marwitz (**Erickson Contractors**)

Great job on this! Not only have you improved service for our customers, but the significant decrease in processing time makes things much simpler for all of us in IT.

From: Nary Loganathan (**Erickson Contractor**)

To: Doug Erickson

And the icing on the cake, tabular reserves completed in 11 hours yesterday night (ran for ~ 4.5 days in previous quarters) ... minor read statement change.

© 2004 John A. Zachman, Zachman International

# Enterprise Architecture

# Conclusions



Plan A (roughly)

SCOPE (CONTEXTUAL)	DATA (What)	FUNCTION (How)	NETWORK (Where)	PEOPLE (Who)	TIME (When)	MOTIVATION (Why)	SCOPE (CONTEXTUAL)
<b>Planner</b> Business Model (CONCEPTUAL) Entry = Class of Business Thing e.g. Semantic Model e.g. Business Process Model	Process = Business Process e.g. Business Process Model	Process = Business Process e.g. Business Process Model	Node = Major Business Location e.g. Business Logistics	Major Organization Unit e.g. Work Flow Model	End/Start Business Event/Queue e.g. Business Plan	End/Start Business Event/Queue e.g. Business Plan	<b>Planner</b> Business Model (CONCEPTUAL)
<b>Owner</b> Ent. = Business Entity Rel. = Business Relation e.g. Logical Data Model	TRD = Business Process ICD = Business Relation e.g. Application Archicture	TRD = Business Process ICD = Business Relation e.g. Application Archicture	Node = Business Location Link = Business Location e.g. Distributed System	People = Organization Unit Work = Work Product e.g. Human Interface	Ent. = Business Event Ment. = Business Event e.g. Processing Structure	End = Business Objective Ment. = Business Strategy e.g. Business Rule Model	<b>Owner</b> System Model (LOGICAL)
<b>Designer</b> Ent. = Data Entry Rel. = Data Relationship e.g. Physical Data Model	Proc = Application Function IO = User View e.g. System Design	Proc = Application Function IO = User View e.g. System Design	Node = IS Structure Link = Technology Architecture e.g. Network Architecture	People = Role Work = Deliverable e.g. Security Architecture	System Event Queue = Processing Cycle e.g. Rules Based	End = Structural Assesment Ment. = Action Assesment e.g. Risk Assesment	<b>Designer</b> System Model (LOGICAL)
<b>Builder</b> Ent. = Task/Requirement, etc. Rel. = Key/Pointer, etc. e.g. Data Definition	Proc = Computer Function IO = Data/Information e.g. Program	Proc = Computer Function IO = Data/Information e.g. Program	Node = Hardware/Software Link = Line of Code/Code e.g. Network Archicture	People = User Work = System Format e.g. Security Architecture	Time = Execute Cycle = Component Cycle e.g. Timing Diagram	End = Condition Ment. = Action e.g. Rules Specification	<b>Builder</b> Technology Model (PHYSICAL)
<b>Sub-Contractor</b> Ent. = Part Rel. = Address e.g. Allocation	Proc = Language Statement IO = Control Block e.g. Allocation	Proc = Language Statement IO = Control Block e.g. Allocation	Node = Address Link = Protocol e.g. Allocation	People = Identity Work = Job e.g. Organization	Time = Interrupt Cycle = Machine Cycle e.g. Scheduler	End = Sub-condition Ment. = Step e.g. Priority	<b>Sub-Contractor</b> Representation (OUT-OF-CONTEXT) Functioning Enterprise

Top Down - Do It Right Approach

## Enterprise Management

My advice to the General Manager:

First, you don't have to know how to build all these models before you can manage the Enterprise (any more than you have to know how to engineer a building before you can occupy it) ... BUT ... I strongly suggest that:

A. **YOU** should make the long term/short term architecture trade-off decision yourself.

Do you want to optimize the *Enterprise*?

Or, do you want to optimize the *implementation* -  
(and *sub-optimize* the Enterprise ... Remember entropy.)

(I would NOT let a programmer  
make the decision by default!)

B. **YOU** decide who should be involved in defining what business you are in and in engineering your Enterprise for you. (That is, who should be involved in taking the assumptions out of the Rows 1 and 2 models for you if you are not going to do it yourself.)

© 2000 John A. Zachman, Zachman International

## Enterprise Management

- C. You ensure that your Enterprise acquires the skills, capabilities and tools to build at least the Rows 1, 2, and 3 models and to maintain control of those models in house whether you use contractors to help build them or not and whether you contract out the production and mgmt of some portions of the Rows 4 and 5 models or not.
- D. You expect your Information Management to analyze your Enterprise Architecture strategy in the context of the Framework and communicate to you the alternatives and implications in the same context.
- E. You establish an Executive Vice President in charge of Change Management to:
1. Define and manage the Enterprise Change Process.
  2. Define and implement Plans and Controls for Architecture Management.
  3. Develop the Architecture Strategy to achieve your long term vision.
  4. Skew the Enterprise culture (including I/S) toward the long term end of the spectrum.

© 2000 John A. Zachman, Zachman International

# Enterprise Management

F. Then, I would capitalize on your ability

to manage complexity  
and  
to dynamically change the Enterprise  
(with minimum time, disruption and cost)

by

destabilizing the market.

P.S. You probably would find it helpful to have at least a passing knowledge about what the Enterprise Architecture models are expressing for you and how to use them in your management of the Enterprise. You would be better positioned to make the short term/long term trade-off decisions and also have a capability for being creative about how the Enterprise could be optimized and changed.

© 2000 John A. Zachman, Zachman International